

# Distributed Optimization in Wireless Networks Using Broadcast Advantage

Tao Cui, Lijun Chen, and Tracey Ho

**Abstract**—In this paper, we consider cross layer optimization in wireless networks with wireless broadcast advantage, focusing on the problem of distributed scheduling of broadcast links. The wireless broadcast advantage is most useful in multicast scenarios. For a multicast scenario, we give a subgradient algorithm for distributed joint congestion control, network coding and session scheduling, which however requires centralized link scheduling. Under the primary interference model, link scheduling problem is equivalent to a maximum weighted hypergraph matching problem that is NP-complete. To solve the scheduling problem distributedly, locally greedy and randomized approximation algorithms are proposed and shown to have bounded worst-case performance. With random network coding, we obtain a fully distributed cross-layer design. Numerical results show promising throughput gain using the proposed algorithms, and surprisingly, in some cases even with less complexity than cross-layer design without broadcast advantage.

## I. INTRODUCTION

Optimization-based cross-layer design for wireless networks has attracted much interest recently, see, e.g., [1]–[4] and the references therein. Joint optimization of multiple protocol layers can substantially increase the end-to-end throughput, or reduce power consumption. Most existing works on cross-layer design do not incorporate exploitation of the wireless broadcast advantage where transmissions from an omnidirectional antenna can be received by any nodes that lie within its communication range. This broadcast advantage can result in power saving and throughput improvement especially with multicasting [5]. In this paper we consider distributed techniques for wireless link scheduling that take the broadcast advantage into account. We apply this to a distributed joint optimization of multicast network coding, congestion control, and medium access.

We model the wireless network as a directed hypergraph, with wireless broadcast being abstracted as a hyperarc. Scheduling with broadcast advantage is a hard problem in general. It forms a component of the algorithm proposed in [6], where it is assumed to be solved by a central controller. In this paper we focus on a simpler primary interference model [7]. Under this interference model, we find that any valid link schedule corresponds to a *hypergraph matching* and the optimal schedule corresponds to a *maximum weighted hypergraph matching*.

The maximum weighted hypergraph matching problem is NP-complete [8]. We propose two classes of distributed

approximation algorithms to solve the link scheduling problem under primary interference model. The first class of algorithms is locally greedy algorithm, which chooses the locally heaviest hyperedge. We show this algorithm returns a hypergraph matching with weight at least a constant factor of the maximum weighted hypergraph matching. The second class of algorithms is randomized algorithm, which always returns a maximal hypergraph matching. The rate stability region with randomized algorithm is shown to be at least  $1/K$  of that with the maximum weighted hypergraph matching, where  $K$  is the maximum number of nodes in any hyperedge. The randomized algorithm can be readily turned into a constant-time algorithm.

We also provide a generalization of existing results in cross layer optimization for multicast network coding in wireless networks. Our cross-layer design uses the framework of utility maximization, see, e.g., [4]. Our objective is to maximize the aggregate user utilities subject to the flow conservation on the hypergraph. We then apply duality theory to decompose the problem vertically into congestion control, network coding and session scheduling, and link scheduling subproblems, which interact through dual variables. Based on this decomposition, a distributed subgradient algorithm is proposed, whose session and link scheduling components are similar to the back-pressure algorithm in [6] which does not incorporate rate control.

**Related Work:** The primary interference model was introduced in [7]. Tassiulas [9] studied randomized algorithms that achieve the capacity region with reduced complexity by comparing a random matching and the current matching. In [10], a distributed implementation of the algorithm in [9] is proposed to achieve the entire capacity region. Various scheduling algorithms based on maximal matching are also proposed or used in, e.g., [2], [6]. All those works do not use the broadcast advantage. Various works have considered optimization with network coding, see, e.g., [11]–[14].

## II. PRELIMINARIES

### A. Network Model

A wireless network is modeled as a directed hypergraph  $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ , where  $\mathcal{N}$  is the set of nodes and  $\mathcal{A}$  is the set of hyperarcs. A hyperarc is a pair  $(i, J)$ , with  $i \in \mathcal{N}$  the start node and  $J \subseteq \mathcal{N}$  the set of end nodes, representing a broadcast link from node  $i$  to nodes in  $J$ . We assume that  $(i, J)$  is lossless, i.e., it does not experience packet erasures. When  $J$  only contains a single node  $j$ , the hypergraph reduces to the conventional graph model. A set  $\mathcal{M}$  of multicast sessions is transmitted through the network. Each session  $m \in \mathcal{M}$  is associated with a set  $\mathcal{S}_m \subset \mathcal{N}$  of sources

This work has been supported in part by DARPA grant N66001-06-C-2020, Caltech's Lee Center for Advanced Networking, a gift from Microsoft Research, and NSF through grant CNS-0435520.

T. Cui, L. Chen, and T. Ho are with Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA, USA 91125. Email: {taocui@, chen@cds., tho@}caltech.edu.

and a set  $\mathcal{T}_m \subset \mathcal{N}$  of sinks. In session  $m$ , each source  $s \in \mathcal{S}_m$  multicasts  $x^{ms}$  bits per second to all the sinks in  $\mathcal{T}_m$ . By the flow conservation condition,

$$\sum_{\{J|(i,J) \in \mathcal{A}\}} \sum_{j \in J} g_{iJj}^{mst} - \sum_{j \in \mathcal{N} \setminus \{i\}} \sum_{\{I|(j,I) \in \mathcal{A}, i \in I\}} g_{jIi}^{mst} = \sigma_i^{ms}, \forall i, \quad (1)$$

where  $\sigma_i^{ms} = x^{ms}$  if  $i = s$ ,  $\sigma_i^{ms} = -x^{ms}$  if  $i = t$ ,  $\sigma_i^{ms} = 0$  otherwise, and  $g_{iJj}^{mst}$  is the rate from source  $s$  to sink  $t$  in session  $m$  over  $(i, J)$  and is intended to node  $j \in J$ .

Let  $\underline{S}(t) = \{S_{i,j}(t)\}$  denote the matrix process of channel states, where  $S_{i,j}(t)$  represents the channel state from node  $i$  to node  $j$  at time  $t$ . Every time slot, node  $i$  determines transmission rates on each hyperarc  $(i, J) \in \mathcal{A}$  by allocating a power matrix  $\underline{P} = (P_{iJ})$  subject to a total power constraint

$$\sum_{(i,J) \in \mathcal{A}} P_{iJ} \leq P_i^{\text{tot}}, \forall i \in \mathcal{N}, \quad (2)$$

where  $P_i^{\text{tot}}$  is the total power at node  $i$ . Hyperarc rates are determined by a rate-power curve  $\underline{r}(\underline{P}, \underline{S}) = \{r_{iJ}(\underline{P}, \underline{S})\}$ , where  $r_{iJ}(\underline{P}, \underline{S})$  determines the rate at which packets, injected into hyperarc  $(i, J)$ , are received by all the nodes in  $J$ . By time-sharing, the capacity region is the convex hull  $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$  of all achievable rate vectors  $\underline{r}(\underline{P}, \underline{S})$ .

### B. Network Coding

In network coding, nodes are allowed to perform algebraic operations on received packets. Network coding promises efficient information transfer over networks. We assume that coding is done only across packets of the same session. With this setting, we define  $f_{iJ}^m$  as the physical flow of session  $m$  on hyperarc  $(i, J)$  as opposed to the virtual flow  $g_{iJj}^{mst}$  in (1). By the flow sharing property of network coding and rate constraints, we have the following two constraints

$$\sum_{s \in \mathcal{S}_m} \sum_{j \in J} g_{iJj}^{mst} \leq f_{iJ}^m, \quad \forall (i, J) \in \mathcal{A}, m \in \mathcal{M}, t \in \mathcal{T}_m, \quad (3)$$

$$\sum_{m \in \mathcal{M}} f_{iJ}^m \leq r_{iJ}, \quad \forall (i, J) \in \mathcal{A}, \quad (4)$$

where  $\{r_{iJ}\}$  belongs to  $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ . To ensure fully distributed cross-layer design, we will use distributed random network coding, see, e.g., [15].

### III. CROSS-LAYER DESIGN WITH BROADCAST ADVANTAGE AND NETWORK CODING

In this section, we derive the cross-layer design by using utility maximization framework, which are extensions of [6], [14]. Each source  $s$  of session  $m$  is associated with a utility function  $U_{ms}(x^{ms})$ , which is assumed to be strictly concave, non-decreasing and twice continuously differentiable. Our objective is to choose source rates  $x^{ms}$  so as to solve the following problem

$$\begin{aligned} \max_{x, g, f, r, P} \quad & \sum_{m \in \mathcal{M}, s \in \mathcal{S}_m} U_{ms}(x^{ms}) \\ \text{s.t.} \quad & \sum_{\{J|(i,J) \in \mathcal{A}\}} \sum_{j \in J} g_{iJj}^{mst} - \sum_{j \in \mathcal{N} \setminus \{i\}} \sum_{\{I|(j,I) \in \mathcal{A}, i \in I\}} g_{jIi}^{mst} = \sigma_i^{ms}, \\ & \sum_{s \in \mathcal{S}_m, j \in J} g_{iJj}^{mst} \leq f_{iJ}^m, \quad \forall (i, J), m, t, \\ & \sum_{m \in \mathcal{M}} f_{iJ}^m \leq r_{iJ}, \quad \forall (i, J), \\ & (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})), \quad \sum_{(i,J) \in \mathcal{A}} P_{iJ} \leq P_i^{\text{tot}}, \quad \forall i, \end{aligned} \quad (5)$$

where  $\sigma_i^{ms}$  is defined in (1), and the constraints come from (1)-(4). Problem (5) is strictly convex and has a unique solution with respect to source rates  $x^{ms}$ . The partial dual function to (5), by relaxing only the first set of constraints, can be decomposed into two subproblems

$$\begin{aligned} \phi_1(q) &= \max_x \sum_{m,s} U_{ms}(x^{ms}) - \sum_{m,s} \left( \sum_t q_s^{mst} \right) x^{ms}, \quad (6) \\ \phi_2(q) &= \max_{g, f, r, P} \sum_{i,m,s,t} q_i^{mst} \left( \sum_{\{J|(i,J) \in \mathcal{A}\}} \sum_{j \in J} g_{iJj}^{mst} \right. \\ &\quad \left. - \sum_{j \in \mathcal{N} \setminus \{i\}} \sum_{\{I|(j,I) \in \mathcal{A}, i \in I\}} g_{jIi}^{mst} \right), \quad (7) \\ \text{s.t.} \quad & \sum_{s,j} g_{iJj}^{mst} \leq f_{iJ}^m, \quad \sum_m f_{iJ}^m \leq r_{iJ}, \\ & (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})), \quad \sum_{(i,J)} P_{iJ} \leq P_i^{\text{tot}}, \end{aligned}$$

where  $q_i^{mst}$  is the Lagrange multiplier at node  $i$  for source  $s$  and sink  $t$  in session  $m$ . The first subproblem is rate control. The second one is the joint network coding and scheduling. Thus, by dual decomposition, the flow optimization problem decomposes into separate “local” optimization problems of transport, and network/link layers, respectively. The two subproblems interact through the dual variable  $q$ .

**Rate Control:** At time  $\tau$ , given dual variable  $q(\tau)$ , each source adjusts its sending rate according to the aggregate dual variables  $\sum_t q_s^{mst}$  that is generated locally at the source

$$x^{ms}(\tau + 1) = U_{ms}^{\prime-1} \left( \sum_t q_s^{mst}(\tau) \right). \quad (8)$$

**Session Scheduling and Network Coding:** Note that (7) is equivalent to the following problem

$$\begin{aligned} \max_{g, f, r, P} \quad & \sum_{(i,J), m, t, s, j \in J} g_{iJj}^{mst} (q_i^{mst} - q_j^{mst}), \\ \text{s.t.} \quad & \sum_{s, j \in J} g_{iJj}^{mst} \leq f_{iJ}^m, \quad \sum_m f_{iJ}^m \leq r_{iJ}, \\ & (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})), \quad \sum_{(i,J)} P_{iJ} \leq P_i^{\text{tot}}, \end{aligned} \quad (9)$$

$$= \max_{f, r, P} \sum_{(i,J), m} f_{iJ}^m \sum_t \max_{s, j \in J} [q_i^{mst} - q_j^{mst}]^+,$$

$$\text{s.t.} \quad \sum_m f_{iJ}^m \leq r_{iJ}, \quad (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})), \quad \sum_{(i,J)} P_{iJ} \leq P_i^{\text{tot}},$$

where  $[\cdot]^+$  denotes the projection onto  $\mathbb{R}^+$ . The last equality in (9) comes from the fact that  $\sum_{s, j \in J} g_{iJj}^{mst} (q_i^{mst} - q_j^{mst})$ , subject to  $\sum_{s, j \in J} g_{iJj}^{mst} \leq f_{iJ}^m$  is a linear programming, we can always choose an extreme point solution, i.e.,

$$g_{iJj}^{mst} = \begin{cases} f_{iJ}^m, & \text{if } s = \hat{s}^{mt}, j = \hat{j}^{mt}, \text{ and } q_i^{mst} - q_j^{mst} \geq 0, \\ 0, & \text{otherwise,} \end{cases} \quad (10)$$

where  $\{\hat{s}^{mt}, \hat{j}^{mt}\} = \arg \max_{s, j \in J} (q_i^{mst} - q_j^{mst})$ .

Let  $\hat{m}_{iJ} = \arg \max_m \sum_t \max_{s, j \in J} [q_i^{mst} - q_j^{mst}]^+$  be the session with the maximum aggregate differential link prices over hyperarc  $(i, J)$ . For each hyperarc  $(i, J)$ , a random linear combination of packets from sources  $\hat{s}^{\hat{m}_{iJ}t}$ ,  $\forall t \in \mathcal{T}_{\hat{m}_{iJ}}$ , in session  $\hat{m}_{iJ}$  is broadcast to all nodes in  $J$  at the rate of  $r_{iJ}$ , where the packets received by node  $j^{\hat{m}_{iJ}t}$  are intended for sink  $t$  in session  $\hat{m}_{iJ}$ . This is equivalent to solving (7) by

$$g_{iJj}^{mst}(q) = \begin{cases} r_{iJ}, & \text{if } m = \hat{m}_{iJ}, s = \hat{s}^{mt}, j = \hat{j}^{mt}, \\ & \text{and } \max_{s,j \in J} [q_i^{mst} - q_j^{mst}]^+ > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

*Link Scheduling and Power Control:* Define  $w_{iJ} = \max_m \sum_t \max_{s,j \in J} [q_i^{mst} - q_j^{mst}]^+$ . The joint link scheduling and power control problem becomes

$$\max_{r,P} \sum_{(i,J) \in \mathcal{A}} w_{iJ} r_{iJ}, \text{ s.t. } (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})), \sum_{(i,J)} P_{iJ} \leq P_i^{\text{tot}}. \quad (12)$$

Depending on the rate-power function  $r(\cdot, \cdot)$ , the joint link scheduling and power control problem (12) is usually a difficult global optimization problem. In some cases, this optimization problem does not have a polynomial-time solution. In Section IV, we will discuss a special interference model such that (12) can be solved distributedly in polynomial time.

*Dual Variable Update:* Let  $q(\tau)$  denote the dual variable  $q$  at time  $\tau$ . Each node  $i$  updates its dual variable  $q$  as  $q_i^{mst}(\tau + 1) =$

$$\begin{cases} q_i^{mst}(\tau) + \gamma_\tau \left( x^{ms}(\tau) - \sum_{\{J|(i,J) \in \mathcal{A}\}} \sum_{j \in J} g_{iJj}^{mst}(q(\tau)) \right. \\ \quad \left. + \sum_{j \in \mathcal{N}} \sum_{\{(j,I) \in \mathcal{A}, i \in I\}} g_{jIi}^{mst}(q(\tau)) \right), & \text{if } i = s, \\ q_i^{mst}(\tau) + \gamma_\tau \left( \sum_{j \in \mathcal{N}} \sum_{\{(j,I) \in \mathcal{A}, i \in I\}} g_{jIi}^{mst}(q(\tau)) \right. \\ \quad \left. - \sum_{\{J|(i,J) \in \mathcal{A}\}} \sum_{j \in J} g_{iJj}^{mst}(q(\tau)) \right), & \text{otherwise,} \end{cases} \quad (13)$$

where  $\gamma_\tau$  is a positive stepsize. After node  $i$  updates  $q_i^{mst}$ , it passes the value to all its neighbors for next time slot rate control, scheduling and network coding. Note that (8)-(13) only requires nodes to communicate with neighbors.

By using results on the convergence of the subgradient method, we can show that, for constant stepsize, our algorithm converges to within a small neighborhood of the optimum as in, e.g., [1]. We omit the proof here for brevity.

#### IV. LINK SCHEDULING

In this section, we study the joint link scheduling and power control problem (12) for networks with primary interference. A system is stable if the queue lengths at all nodes remain finite. A rate vector  $\vec{x} = \{x^{ms}\}$  is feasible if there exists a scheduling policy that stabilizes the system with  $\vec{x}$ . We define the capacity region  $\Lambda$  to be the set of all feasible rate vectors. We are interested in scheduling policies that can stabilize the system for any rate vector within  $\gamma\Lambda$ , where  $\gamma \in (0, 1]$  is a constant that characterizes the performance of the policy. The scheduling algorithm and nodes' coordination are performed at the beginning of each time slot.

##### A. Problem Formulation

Under the primary interference model, only those links that do not share nodes can transmit at the same time. It models a situation where each node is equipped with a single

<sup>1</sup>The queue length of node  $i$  at time  $t$  can be written as  $q_i^{mst}/\gamma_t$  for a constant stepsize  $\gamma_t$  [2].

transceiver and neighboring nodes can transmit simultaneously using orthogonal CDMA or FDMA channels. Under this interference model, without using broadcast advantage, any feasible schedule corresponds to a matching. With the broadcast advantage, (12) reduces to maximum weighted hypergraph matching<sup>2</sup> problem.

We assume the use of CDMA in the following. Assuming orthogonal spreading sequences and white Gaussian noise channels, the maximum achievable data rate on  $(i, J)$  is

$$r_{iJ}(\underline{P}, \underline{S}) = \frac{1}{G} \log \left( 1 + \min_{j \in J} \text{SNR}_{i,j} \right), \quad (14)$$

where  $G$  is the spreading factor,  $\text{SNR}_{i,j} = P_i^{\text{tot}} \frac{|h_{i,j}|^2}{\sigma_j^2}$  is the effective SNR from node  $i$  to node  $j$ ,  $\sigma_j^2$  is additive white Gaussian noise power at node  $j$ , and  $h_{i,j}$  is the channel fading coefficient from node  $i$  to node  $j$ . Note that at each time slot only one hyperarc per node is active. Therefore, any feasible schedule corresponds to a hypergraph matching of the hypergraph  $\mathcal{H}$ . Let  $\Pi$  denote the set of all hypergraph matchings. We represent a hypergraph matching  $\pi$  as an  $|\mathcal{A}|$ -dimensional rate vector  $\xi^\pi$

$$\xi_{iJ}^\pi = \begin{cases} r_{iJ}(\underline{P}, \underline{S}), & \text{if } (i, J) \in \pi, \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

The achievable rate region  $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$  is then written as

$$\text{Co}(\underline{r}(\underline{P}, \underline{S})) \triangleq \left\{ \mathbf{r} : \mathbf{r} = \sum_{\pi \in \Pi} \alpha_\pi \xi^\pi, \alpha_\pi \geq 0, \sum_{\pi \in \Pi} \alpha_\pi = 1 \right\}. \quad (16)$$

Note that  $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$  is a polytope. So, we can always pick up an extreme point maximizer for the scheduling problem (12), which corresponds to a maximum weighted hypergraph matching in  $\mathcal{H}$ .

We first transform the directed hypergraph to an equivalent undirected hypergraph  $\tilde{\mathcal{H}} = (\mathcal{V}, \mathcal{E}_h)$ , where  $\mathcal{H}$  and  $\tilde{\mathcal{H}}$  have the same node set. Note that hyperarcs  $(i, J)$  and  $(j, I)$  mutually interfere and have the same interference/contention relations with other hyperarcs if  $\{i\} \cup J = \{j\} \cup I$ . Define an undirected hyperedge  $e \subseteq \mathcal{V}$  in  $\mathcal{E}_h$ , which corresponds to all hyperarcs  $(i, J)$  such that  $e = \{i\} \cup J$ . The weight of hyperedge  $e$  is

$$\tilde{w}_e = \max_{\{(i,J) \in \mathcal{A}, \{i\} \cup J = e\}} w_{iJ} r_{iJ}(\underline{P}, \underline{S}). \quad \forall e \in \mathcal{E}_h, \quad (17)$$

The problem (12) is then equivalent to the maximum weighted hypergraph matching (or maximum weighted set packing) problem on the weighted hypergraph  $\tilde{\mathcal{H}}$ .

Different from the maximum weighted matching problem on graphs which can be computed in polynomial time, the maximum weighted hypergraph matching problem is NP-complete [8]. Also, we would like distributed algorithms. Both factors suggest that we should focus on approximation algorithms.

Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be an undirected graph with the same node set as  $\mathcal{H}$ . We assume that there exists an edge between node  $i$  and node  $j$  only if  $\min\{\text{SNR}_{ij}, \text{SNR}_{ji}\} \geq \gamma$ , where  $\gamma$  is a predefined threshold. This means that if  $i$  can hear  $j$ , then  $j$  can hear  $i$ . Let  $N(v)$  denote the neighbor node set of node  $v$  in  $\mathcal{G}$ . We call  $\mathcal{G}$  *connectivity graph* in the following.

<sup>2</sup>A hypergraph matching is defined as a set of hyperarcs with no pair incident to the same node.

### B. Local Optimal Algorithm

A linear time approximation algorithm with bounded worst-case performance for maximum weighted graph matching is proposed in [16], which adds a locally optimal edge into the matching at each step. Motivated by [16], our algorithm adds a locally heaviest hyperedge into the hypergraph matching at each step.

**Definition 1 (locally heaviest hyperedge):** A hyperedge  $e$  is a locally heaviest hyperedge if its weight is at least as large as the weight of all adjacent hyperedges, i.e.,  $\tilde{w}_e \geq \tilde{w}_f$ ,  $\forall e \cap f \neq \emptyset$ .

```

DLOHMA: ( $\mathcal{G}$ )
1 for each node  $i \in \mathcal{V}$  do
2   Broadcast the set  $\{\text{SNR}_{ij} | j \in N(i)\}$  to all its
   neighbor nodes ;
3   Set  $C_i = \emptyset$ ,  $\Gamma_i = N(i)$ , and  $\Gamma_j^{(i)} = N(j)$ ,
    $\forall j \in N(i)$ ;
4 end
5 for each node  $i \in \mathcal{V}$  do
6   Find a node set  $J^*$  by  $J^* = \{j^*\} \cup L^* - i$  where
    $j^*, L^*$  are obtained via
       
$$(j^*, L^*) = \arg \max_{j \in \Gamma_i \cup \{i\}} \max_{\{L | L \subseteq \Gamma_j^{(i)}, i \in L\}} w_{jL} r_{jL}(\underline{P}, \underline{S}), \quad (18)$$

       and  $w_{jL}, r_{jL}$  are defined in (12) ;
7   if  $J^* \neq \emptyset$  then Broadcast a matching  $e_i^* = \{i\} \cup J^*$ 
   message;
8 end
9 while  $\exists i, \Gamma_i \neq \emptyset$  do
10  if node  $i$  receives a message  $m$  which is has not
   received then
11    switch  $m$  do
12      case matching  $e$ 
13         $C_i(e) = C_i(e) + 1$ ;
14      end
15      case drop  $e$ 
16        if  $i \notin e$  then
17          if  $e \cap \Gamma_i \neq \emptyset$  then Broadcast a drop
           $e$  message;
          Remove the nodes in  $e$  from  $\Gamma_i$  and
          all  $\Gamma_j^{(i)}, j \in \Gamma_i$ ;
          if  $e \cap J^* \neq \emptyset$  then
18            Find a node set  $J^*$  by (18);
            if  $J^* \neq \emptyset$  then Broadcast a
            matching  $e_i^* = \{i\} \cup J^*$ 
            message;
19          end
20        else if  $\Gamma_i \neq \emptyset$  then Broadcast a drop  $e$ 
        message, and set  $\Gamma_i = \emptyset$ ;
21      end
22    end
23  if  $J^* \neq \emptyset$  and  $C_i(e_i^*) = |J^*|$  then
24    Broadcast a drop  $e_i^*$  message, and set
     $\Gamma_i = \emptyset$ ;
25  end
26 end
27 end
28 end
29 end
30 end

```

**Algorithm 1:** Distributed local optimal algorithm.

The distributed local optimal hypergraph matching algorithm is given in Algorithm 1. In Algorithm 1, the set  $\Gamma_i$  maintains the set of neighbors of node  $i$  that are still not matched, which is initialized to be all its neighbors in  $\mathcal{G}$ . Node  $i$  also maintains the neighbor set  $\Gamma_j^{(i)}$  for each neighbor

$j$  to facilitate the computation of  $\tilde{w}_e$  in (17). The vector  $C_i$  counts the number of matching  $e_i^*$  messages that have been received, which is initialized to be a null vector (line 3). Each node  $i$  broadcasts a matching  $e_i^*$  message, where  $e_i^* = \{i\} \cup J^*$  is the maximum hyperedge in  $\mathcal{H}$  containing  $i$  (lines 5-8). If node  $i$  receives  $|J^*|$  matching  $e_i^*$  messages, hyperedge  $e_i^*$  is added to the hypergraph matching as  $e_i^*$  is a locally heaviest hyperedge. It broadcasts a drop  $e_i^*$  message to indicate that  $i$  is matched and unavailable, and at the same time to tell all nodes in  $e_i^*$  that they are matched (lines 26-28). If node  $i$  receives a drop  $e$  message and node  $i$  is not in  $e$ , it first checks whether some nodes of  $e$  are in  $\Gamma_i$ . If yes,  $i$  is the direct neighbor of some nodes in  $e$  and  $i$  broadcasts drop  $e$  message to let  $i$ 's neighbors (two-hop neighbor of the nodes in  $e$ ) know that all the nodes in  $e$  are matched. If not, some nodes in  $e$  are two-hop neighbors of  $i$  and we do not need to forward the drop message. Node  $i$  then removes the nodes in  $e$  from  $\Gamma_i$  and all  $\Gamma_j^{(i)}, j \in \Gamma_i$ . Furthermore, if some nodes in  $J^*$  are in  $e$ , the hyperedge  $e_i^*$  is dropped. Node  $i$  then finds another candidate set  $J^*$ , and it broadcasts a new matching  $e_i^*$  message (lines 16-23). If  $i$  receives a drop  $e$  message and node  $i$  is in  $e$ ,  $i$  will broadcast a drop  $e$  message if it did not do so before, i.e.,  $\Gamma_i$  is nonempty (line 23).

Note that some nodes in the locally heaviest hyperedge may not be able to hear each other. These nodes cannot receive  $|J^*|$  matching  $e$  messages and conclude that  $e$  is the locally heaviest hyperedge. But at least one node can hear all the other nodes in the hyperedge. This is the reason why we broadcast a drop  $e$  message in line 27.

In Algorithm 1, we assume that all hyperedges have different weights. If they do not, we can always break the tie by adding a small constant  $\epsilon_e$  to  $w_e$  (different  $e$  has different  $\epsilon_e$ ). For example, we can change  $w_{iJ}$  or  $r_{iJ}$  by a small constant. In the following, we also assume that the cardinality of all hyperedges in  $\mathcal{E}_h$  is bounded from above by a constant  $K$ . Let  $\kappa = \max_{m \in \mathcal{M}} |\mathcal{T}_m| + 1$ . Due to space limitation, we omit all the proofs of propositions and theorems in the following. The proofs can be found in [17].

**Proposition 1:** The hyperedge  $e_i^*$  in line 26 is a locally heaviest hyperedge.

**Proposition 2:** In Algorithm 1, each node  $i$  broadcasts at most  $\sum_{j \in N(i)} |N(j)| + |N(i)|$  messages.

Different from [16] where node  $i$  only sends a message to node  $j$ , we make use of the broadcast property of wireless communication, which reduces the number of messages.

**Theorem 1:** The complexity of Algorithm 1 is  $O\left(K^3 |\mathcal{E}| \sum_{k=1}^{\min\{\kappa, K\}-1} \binom{K-1}{k}\right)$  time and the number of time-slots required to finish Algorithm 1 is  $O(|\mathcal{V}|)$ .

**Theorem 2:** Algorithm 1 computes a hypergraph matching  $HM_{LO}$  with at least  $\max\{\frac{1}{K}, \frac{1}{\kappa}\}$  of the weight of a maximum weighted hypergraph matching  $HM_{MW}$ .

In Algorithm 1, some matched nodes may not contribute much to this locally heaviest hyperedge. But when these nodes are matched in other hyperedges, they may contribute more, which results in a hypergraph matching with

higher weight. Instead of choosing the hyperedge according to its weight, we use the average hyperedge weight, i.e.,  $\bar{w}_e = \bar{w}_e/|e|$ . We modify Algorithm 1 to **Algorithm 2** by simply replacing  $\bar{w}_e$  with  $\bar{w}_e$ . Both the complexity and the approximation ratio of Algorithm 2 are identical to those of Algorithm 1.

**Theorem 3:** Both Algorithm 1 and Algorithm 2 stabilize the system for any rate vector  $\vec{x}$  such that  $\vec{x} + \epsilon \in \max\{\frac{1}{K}, \frac{1}{\kappa}\}\Lambda$ .

This theorem can be shown by following the proof in [2]. The high complexity of algorithms 1 and 2 is also due to that we need to propagate drop  $e$  message to all two-hop neighbors of the nodes in  $e$ . If we assume that any node can receive drop  $e$  message from its two-hop neighbors, the complexity in Theorem 1 can be decreased by a factor of  $K$ .

### C. Randomized Algorithm

In this subsection, we consider randomized algorithms to find a maximal hypergraph matching. We start with the definition of the maximal hypergraph matching.

**Definition 2 (maximal hypergraph matching):** A hypergraph matching  $HM$  is maximal if for each hyperedge  $e \in \mathcal{H}$ , one or more of the following conditions are satisfied:

- $e \cap HM \neq \emptyset$ , i.e.,  $e$  has non-empty intersection with at least one hyperedge in  $HM$ .
- $\bar{w}_e = 0$  or the number of packets waiting to be transmitted over the hyperedge is zero.

#### DRHMA: ( $\mathcal{G}'$ )

```

1 for each node  $i \in \mathcal{V}$  do Set  $\Gamma_i = N(i)$ ;
2 while  $\exists i, \Gamma_i \neq \emptyset$  do
3   for each node  $i \in \mathcal{V}$  and  $\Gamma_i \neq \emptyset$  do
4     Let  $p$  be a random number generated according
      to the uniform distribution on  $[0, 1]$ .
5     if  $p < \frac{1}{|\Gamma_i|}$  then
6       For each node  $j \in \Gamma_i$ , with probability  $\frac{1}{2}$ 
        add  $j$  into set  $S_i$ ;
7     end
8     if  $S_i \neq \emptyset$  then
9       Node  $i$  decides to transmit and it broadcasts
        matching messages to all nodes in  $S_i$ ;
10      Set  $E_i = \emptyset$ ;
11    end
12  end
13  for each node  $i \in \mathcal{V}$  and node  $i$  does not transmit do
14    if node  $i$  receives matching messages from
      several neighbors then
15      Node  $i$  chooses one of them uniformly at
        random, say  $j$ , and sets  $\Gamma_i = \emptyset$ ;
16      Node  $i$  broadcasts a  $i$  matched  $j$  message;
17    end
18  end
19  while  $\exists k, k$  receives a  $i$  matched  $j$  message do
20    if  $k = j$  then  $E_k = E_k \cup \{i\}$ ;
21    else  $\Gamma_k = \Gamma_k - \{i\}$ ;
22  end
23  for each node  $i \in \mathcal{V}$ , and if  $i$  decides to transmit do
24    if  $E_i \neq \emptyset$  then  $E_i$  is added into the hypergraph
      matching, and set  $\Gamma_i = \emptyset$ ;
25  end
26 end

```

**Algorithm 3:** Distributed randomized algorithm.

The distributed randomized hypergraph matching algorithm is given in Algorithm 3. The input of Algorithm 3 is a graph  $\mathcal{G}'$  is after deleting all the edges  $\{i, j\}$  with both  $\max_{m,s,t} [q_i^{mst} - q_j^{mst}]^+ = 0$  and  $\max_{m,s,t} [q_j^{mst} - q_i^{mst}]^+ = 0$  from  $\mathcal{G}$ , which guarantees that all the hyperedges have positive weights. In Algorithm 3, the set  $\Gamma_i$  maintains the set of neighbors of node  $i$  that are still not matched, which is initialized to be all its neighbors in  $\mathcal{G}$  (line 1). Each unmatched node  $i$  attempts to transmit with probability  $\frac{1}{|\Gamma_i|}$  (line 5). If  $i$  attempts to transmit, for each neighbor  $j$ , it sends a matching request to  $j$  with probability 1/2 (line 6), which is because we would like to pick any hyperedge containing  $i$  with equal probability. If  $i$  sends request to at least one neighbor, i.e.,  $S_i \neq \emptyset$ , it decides to transmit (line 9).  $E_i$  denotes the hyperedge to be added into the matching initialized by  $i$  (line 10). If node  $i$  does not transmit and it receives several matching requests from its neighbors, it chooses one of them uniformly at random, say  $j$ , sets  $\Gamma_i = \emptyset$  ( $i$  is matched), and broadcasts an “ $i$  matched  $j$ ” message (lines 13-18). On receiving an “ $i$  matched  $j$ ” message, node  $k$  checks whether  $k = j$ . If  $k = j$ , this indicates that  $i$  got the matching request from  $k$  and it would like to join in the hyperedge initialized by  $k$ . Thus,  $k$  sets  $E_k = E_k \cup \{i\}$  (line 20). If  $k \neq j$ , this just indicates that  $i$  got matched to  $j$  and  $k$  should delete  $i$  from  $\Gamma_k$  (line 21). For all the nodes that decide to transmit, if finally  $E_i \neq \emptyset$ ,  $E_i$  is added into the hypergraph matching, and we set  $\Gamma_i = \emptyset$  ( $i$  is matched). Algorithm 3 returns a maximal hypergraph matching according to Definition 2.

**Theorem 4:** The expected running time of Algorithm 3 is  $O(\log |\mathcal{E}|)$ .

Compared with Algorithm 1, Algorithm 3 not only reduces the time complexity from  $O(|\mathcal{E}|)$  to  $O(\log |\mathcal{E}|)$  but it also does not need to compute the weight of each hyperedge.

**Theorem 5:** Algorithm 3 stabilizes the system for any rate vector from  $\frac{1}{K}\Lambda$  if in each session all sinks are only one hop away from the source.

Algorithm 3 can be readily turned into a constant-time algorithm by executing the while loop in Algorithm 3 only  $M$  times. We call this algorithm **Algorithm 4**.

As in Section IV-B, we do not consider packet collision during hypergraph matching. Algorithms 3 and 4 can be readily adapted to this case. More algorithms such as a hybrid algorithm and extensions of the graph matching algorithm can be found in [17].

As maximal matching takes an important role in many scheduling algorithms, e.g., [9], [10], we expect that our Algorithm 3 can also serve as a basis for other scheduling algorithms for our problem. Note that the approach in [10] cannot be trivially adopted as the connected components in the union of the new hypergraph matching and the old hypergraph matching may be very large. Also, the connected components are not simply cycles or paths as in [10].

### V. SIMULATION RESULTS

The node  $i$ 's signal power is attenuated by a factor of  $\rho_{i,j}^{-1}$  when the signal is received by node  $j$ , where  $\rho_{i,j}$  is

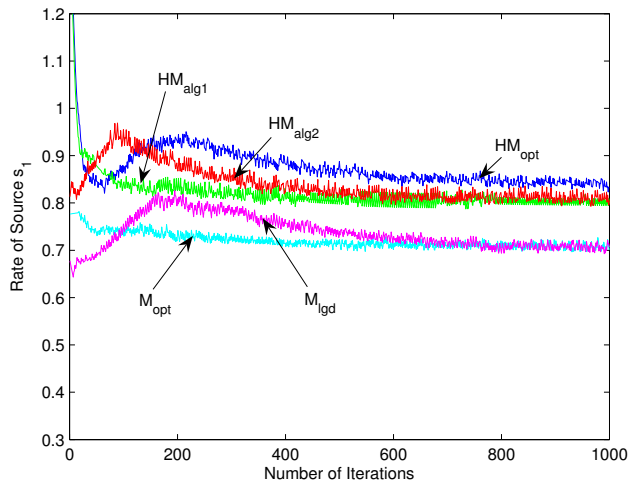


Fig. 1. The evolution of source  $s_1$ 's rate versus the number of iterations with fixed stepsize  $\gamma = 0.01$  for the wireless butterfly network, where maximum weighted hypergraph matching, Algorithms 1 and 2, and maximum weighted graph matching and local greedy matching are compared.

the Euclidean distance between  $i$  and  $j$ . All nodes have unit signal power and identical noise power 0.1. We adopt (14) for computing  $r_{i,j}$ . We neglect the factor  $1/G$  in (14). Two nodes  $i$  and  $j$  are considered to be connected if and only if link  $(i,j)$ 's capacity is at least 1.

We first consider the wireless butterfly network with two sources  $s_1, s_2$ , two sinks  $t_1, t_2$  and one relay node  $r$ , where the coordinates (in meters) of  $s_1, s_2, t_1, t_2, r$  are  $(0, 5), (5, 5), (0, 0), (5, 0)$ , and  $(2.5, 2.5)$ . Each source multicasts data to both sinks. We thus only consider a single multicast session. Fig. 1 shows the evolution of source  $s_1$ 's rate versus the number of iterations with fixed stepsize  $\gamma = 0.01$ , where maximum weighted hypergraph matching ( $\text{HM}_{\text{opt}}$ ), Algorithms 1 and 2 ( $\text{HM}_{\text{alg}i}, i = 1, 2$ ), maximum weighted graph matching ( $\text{M}_{\text{opt}}$ ), and local greedy matching ( $\text{M}_{\text{lgd}}$ ) are compared. The figure for Algorithms 3 and 4 is omitted due to lack of space. We have observed that the rates of Algorithms 3 and 4 oscillate more severely than Algorithms 1 and 2 as the former algorithms use randomized mechanism.  $\text{HM}_{\text{opt}}$  with broadcast advantage has about 20% gain over that without using broadcast advantage. Even with Algorithms 1 and 2, about 14%-15% gain can still be achieved. Algorithm 3, the randomized algorithm, can also achieve a 1.19% gain. Surprisingly, both  $\text{HM}_{\text{alg}1}$  and  $\text{HM}_{\text{alg}2}$  require fewer coordination time-slots than  $\text{M}_{\text{lgd}}$  does, but the former two have higher rates than the latter. This is due to the use of the broadcast advantage during scheduling and that each hyperedge contains several nodes. By changing  $M$  in Algorithm 4, we can see a trade-off between complexity and performance.

We next consider random networks. 10 nodes are randomly and uniformly placed on a 20 meter by 20 meter square. Both source and sinks are randomly chosen. We consider only a single multicast session with one source and 2, 4, and 6 sinks. 1000 feasible network realizations are generated. We observe that  $\text{HM}_{\text{opt}}$  can achieve rate gains 11.86%, 14.07%, and 16.33% for 2, 4, and 6 sinks. Gain increases as the number of sinks increases. The same

observation holds for all the other algorithms. On average, Algorithm 2 performs better than Algorithm 1 as the former takes into account the size of the hyperedge. Our results suggest that it is better to use hypergraph matching when the multicast group is large.

## VI. CONCLUSION

We studied cross-layer optimization for multicasting in wireless networks with wireless broadcast advantage. By designing distributed approximation algorithms for broadcast link scheduling, we gave a fully distributed algorithm for joint congestion control, network coding and scheduling. Numerical results have shown promising throughput gain by using the proposed algorithm, and surprisingly, in some cases with even lower complexity than the cross-layer design without broadcast advantage. Larger gain is expected when power cost is also considered. Our results suggest that broadcast link scheduling can be a promising avenue of further research. For complete proof of theorems and more simulation results, please refer to technical report [17].

## REFERENCES

- [1] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for wireless ad hoc networks," in *Proc. of IEEE Infocom*, Mar. 2005.
- [2] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Networking*, vol. 14, no. 2, pp. 302–315, April 2006.
- [3] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.
- [4] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition," *Proc. of IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.
- [5] J. Wieselthier, G. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. of Infocom*, March 2000, pp. 585–594.
- [6] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," in *Proc. of Allerton Conf. on Comm., Contr. and Comput.*, Sept. 2005.
- [7] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 910–917, Sept. 1988.
- [8] L. Lovasz and M. Plummer, *Matching Theory*. North Holland, 1986.
- [9] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radionetworks and input queued switches," in *Proc. of Infocom*, March 1998, pp. 533–539.
- [10] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 27–38, June 2006.
- [11] D. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.
- [12] Y. Wu and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: A shortest path approach," *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1475–1488, Aug. 2006.
- [13] Z. Li, B. Li, and L. C. Lau, "On achieving maximum multicast throughput in undirected networks," *IEEE/ACM Trans. Networking*, vol. 14, no. SI, pp. 2467–2485, June 2006.
- [14] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Rate control for multicast with network coding," in *Proc. of IEEE Infocom*, 2007.
- [15] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, pp. 4413–4430, Oct. 2006.
- [16] R. Preis, "Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs," in *Proc. of the Symposium on Theoretical Aspects of Computer Science (STACS)*, 1999, pp. 259–269.
- [17] T. Cui, L. Chen, and T. Ho, "Cross-layer design in wireless networks by using broadcast advantage," Caltech, Tech. Rep., Mar. 2007.